Vol.2013-SE-180 No.5
Vol.2013-EMB-29 No.5
2013/5/27

WSNにおけるプログラム位置管理機構の提案と実装

並木 勁汰[1]　福田 浩章[2]

概要：無線センサネットワーク（Wireless Sensor Networks: WSN）は，センサデータを取得できる複数のノードによって構成される無線ネットワークである．WSN を構成するノードは電源が有限であり，省電力を実現するためにノードの計算資源が極小に抑えられている．したがって，複数のアプリケーションを運用する際，必要なノードに対してのみプログラムを配備する必要がある．また，アプリケーションの稼働場所を変更する場合，プログラムの配備場所を変更する必要がある．WSN ではアプリケーションの開発者がプログラムの配備場所を把握した上でプログラムを記述しなければならない．特に，プログラムの移動を考慮しなければならないモバイルエージェントアプローチではそれが困難である．本研究では，WSN におけるプログラム間通信に起因する煩雑さを改善するため，分散ハッシュテーブルによるプログラムの位置管理を行い，プログラムの配備場所を取得できる機構の提案と実装を行った．また，プログラム位置管理機構によるプログラム位置情報探索のシミュレーションを行った結果，比較的少ない消費電力での位置情報取得が可能であることが確認できた．

1. はじめに

無線センサネットワーク（Wireless Sensor Networks: WSN）は，センサを搭載した複数のノードによって構成される無線ネットワークである．物理的にノードを敷設するだけで現実世界の動態を捉えることができる容易さから，幅広い分野で WSN の応用が期待されている．代表的なアプリケーションとして，環境のモニタリング，対象物の検知と追跡，ヘルスモニタリング，構造物のモニタリング等がある．

WSN を構成するノードの特徴として，電力が有限であること，ノードの計算資源が極小であることが挙げられる．WSN では個々のノードが電池で稼働することを想定しており，WSN の最大かつ共通の課題は電源枯渇問題である．そこで，バッテリの限られた電力を節約するため，CPU やメモリ等の計算資源は極小に抑えられている．ゆえに，複数のアプリケーションを運用する際，必要なノードに対してのみプログラムを配備することが重要になる．

WSN を構成する各ノードは，センサ機能を用いて週辺の環境を観察する．観測されたデータは，ノード間のマルチホップ・アドホック通信でベースステーションに集められる．ベースステーションは WSN へのアクセスが可能なコンピュータであり，WSN から得られた環境データを集約・保持している．

1.1 研究の背景

1.1.1 WSN のアーキテクチャ

WSN のアプリケーションを実現するためのアーキテクチャは，大別するとベースステーション集中型とネットワーク内処理型の 2 つが存在する．ベースステーション集中型では各ノードが取得できるセンサデータを定期的にベースステーションに送信し，ベースステーションでセンサデータを解析することによってアプリケーションを実現する．また，ネットワーク内処理型ではアプリケーションを構成するプログラムを各ノードに配備し，必要に応じてベースステーションと通信しながらネットワーク内部でアプリケーションを実行する．

ベースステーション集中型は単純なアーキテクチャである．アプリケーションプログラムはベースステーションで実行されることから，WSN 固有の性質を扱うプログラミング量が比較的少なくて済むという利点がある．しかし，ベースステーション周辺のノードにトラフィックが集中し，電源が枯渇する恐れがある．また，WSN ではパケットロスが発生することが知られており [1]，ベースステーションとの距離が離れたノードや複数ホップを要するノードが存在する場合，データの複数回再送，データの送信・転送ができない可能性がある．したがって，ベースステーションへ頻繁にデータを送信する必要があるベースステーション集中型では，通信の非効率さを誘発する結果となる．

[1] 芝浦工業大学 大学院 理工学研究科 電気電子情報工学専攻
Shibaura Institute of Technology
[2] 芝浦工業大学
Shibaura Institute of Technology

これらのベースステーション集中型の課題を補うため，ネットワーク内処理型のアーキテクチャが用いられる．

1.1.2 WSN のミドルウェア

WSN のプログラミングは制約のあるリソース，動的に変化するネットワーク環境，配備されるノード数，取得可能なデータ等を考慮しなければならない．また，ハードウェアとネットワークに関わる低レイヤーの処理を記述する必要があることから，プログラミングが煩雑になる．そのため，ネットワーク内処理型のアーキテクチャを採用する場合，ノードに配備されるプログラムを一から作成することは非効率であり，ミドルウェアを用いたプログラマの負担軽減が選択肢の一つとなる．WSN におけるミドルウェアの研究は数多く存在するが，大別すると以下の 3 つに分類できる．

1. データベースアプローチ
   計測可能なデータを中心に WSN を抽象化する．WSN の内部動作を意識することなくデータを取得することを可能にしている．主な研究例に TinyDB[2] がある．
2. イベントベースアプローチ
   イベント駆動型の情報処理メカニズムで WSN を抽象化する．把握したい現実世界の状態をイベントとして記述し，イベントが観測された場合のみ結果を通知する．主な研究例に EnviroTrack[3] がある．
3. モバイルエージェントアプローチ
   WSN 内部に配備するプログラムをモバイルエージェントとみなす．個々のノードを渡り歩いて実行する処理を記述することができ，状況に応じた WSN の利用を可能にする．主な研究例に Agilla[4] がある．

WSN 内部で複数のアプリケーションを運用する際，プログラムを動的に配備する手法が有効である．モバイルエージェントアプローチは，資源の限られた一つの WSN において複数のアプリケーションを実現することが可能である．ゆえに，そのような運用形態にモバイルエージェントアプローチは適している．

1.1.3 Agilla

1.1.2 で述べたように，モバイルエージェントアプローチの代表例である Agilla を利用したプログラムは，VM で解釈可能な 1byte〜2byte の ISA（Instruction Set Architecture）で記述され，プログラムのコード容量を削減できる．Agilla のモデルを図 1 に示す．Agilla は，メモリの制約が許す範囲で複数のプログラムを同時にノード上で稼働させること，プログラム間の通信をタプルスペース [5] を用いて記述すること，プログラムの配備場所の指定を行うこと，WSN 内部でプログラムを移動させることを実現している．その結果，プログラマは必要なノードだけにプログラムを配備することや，WSN 内部でのプログラム間通信を必要とするようなアプリケーションを実現することができる．

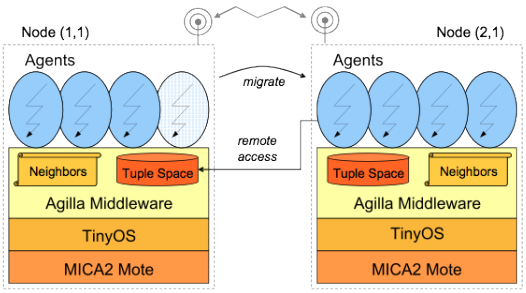


図 1 Agilla のモデル ([4] より引用)

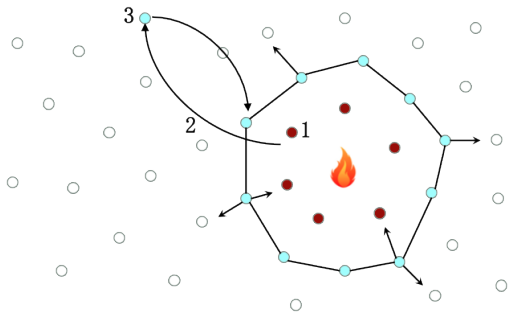


図 2 Agilla のアプリケーション例

1.1.4 Agilla の問題点

Agilla はプログラムの位置管理機構を提供していないため，プログラマがプログラムの配備状況を熟知していなければならない．

図 2 に Agilla での火災探知・追跡を行うアプリケーションの例を示す．火災が発生すると火災探知プログラム（1）が探知し，通知を行う（2）．通知を受けた追跡プログラム（3）が，自身のクローンを火災発生場所周辺のノードへ配備する．そして，3 のプログラムの存在によって火災を囲み，追跡を行うことができる．

このとき，Agilla ではプログラム中に通信対象のノードを指定しなければならないため，図 2 の通信を受ける 3 のようなプログラムは特定のノードに留まる必要がある．通信を受けるプログラムが移動をする場合，ネットワーク内におけるプログラムの所在を記録していないため，通信を行う側のプログラムは相手がどのノードにあるのか探索をしなければならない．ゆえに，WSN 内部を渡り歩くプログラムに対して通信を行うことは困難である．

1.2 本研究の目的

Agilla では，プログラムに通信先のノードを記述する仕様上，WSN 内を渡り歩くプログラムに対する通信は困難であり，また，その仕様はプログラミングの煩雑さを助長している．

そこで，本研究ではモバイルエージェントアプローチにおける，プログラム間通信に起因するプログラミングの煩雑さを改善するため，ミドルウェアの機能として必要

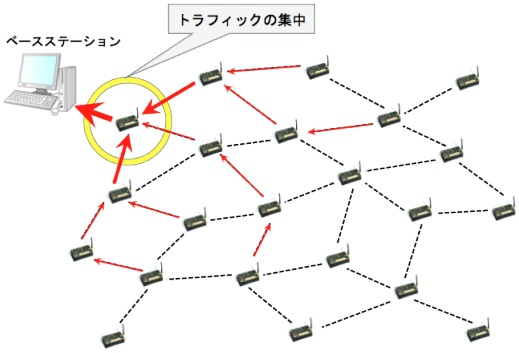


図 3 ベースステーションで管理した場合のトラフィック集中

となる，分散ハッシュテーブル（Distributed Hash Table: DHT）[6] を用いたプログラム位置管理機構の提案と実装を行う．

2. アプローチ

ここではプログラムの位置情報管理を行うにあたり，問題点を明確にし，本研究のアプローチを述べる．また，プログラム位置管理機構の実装に用いる技術を紹介する．

2.1 プログラム位置情報の保管場所

プログラムの位置情報を保管する方法として，ベースステーションで位置情報を保管する手法，WSN 内のノードに保管する手法の 2 つが考えられる．

ベースステーションで保管する場合，保管側の計算資源を考慮する必要がなく，集中管理できるという利点がある．しかし，ベースステーションで情報を集約するという構造上，図 3 に示すように，ベースステーション周辺のノードにトラフィックの集中してしまう．その結果，ベースステーション周辺ノードの電力消費が多くなり，ノードの稼働時間が短くなる．ベースステーションとの通信という WSN における重要な役割を持つノードの稼働時間低下は，WSN 全体の稼働時間低下という重大な問題を発生させる．

そこで，本研究では WSN 内のノードに保管する手法を採る．この手法では，複数のノードに分散してプログラムの位置情報を保管することで，個々のノードに対する負荷の分散が期待できる．本手法によって，ベースステーションでの集中管理で発生する，WSN 全体の稼働時間低下という重大な問題を回避することが可能である．

2.2 位置情報管理の実現方法

WSN におけるノード間の通信は，自身と同じ性能を有するノード同士の通信であり，P2P 通信と考えることができる．そこで，本研究ではプログラムの位置情報を分散管理する手法として，100%に近い探索成功率と非常に高い探索効率を有する DHT を用いる．

DHT には複数のアルゴリズムが存在するが，高速にコンテンツ探索を行える Chord アルゴリズム [7] を採用する．しかし，無線通信を行う WSN の性質上，ノード間の物理的距離を考慮しない Chord をそのまま用いることは適切ではない．したがって，プログラム位置管理機構は Chord をセンサネットワークに適応させた CSN（Chord for Sensor Networks）アルゴリズム [8] を用いて実現する．

2.3 DHT

DHT は，第 3 世代 P2P システムにおいて広く利用されている要素技術である．DHT によって，ディレクトリ・サービス（中央サーバ）を必要としない純粋なピア・ツー・ピア型のネットワーク型分散ファイルシステムを実現することができる．ファイル識別子は，ファイルの名前やファイル自体からハッシュ関数を用いて生成される数値によって表現される．そのため，ファイル名の匿名性を実現しながら，ファイルの位置情報をひとつのピアに偏ることなく分散して管理することができる．

また，DHT は高速にネットワーク内を探索でき，検索に伴うトラフィックの負荷が極めて軽い管理方式である．すべてのオンライン端末（ノード）のコンテンツ検索が可能であり，多数のノードの参加・離脱に耐えることができる．

2.4 Chord

Chord は，DHT を代表するアルゴリズムのひとつである．Chord では，時計回りに識別子の値が増加していくリング状の識別子空間を考え，識別子は SHA1 ハッシュ関数の値を用いる．SHA1 の値は 160 ビットの数値のため，Chord の識別子空間の大きさは 160 ビットになる．

図 4 に識別子空間の大きさを 6 ビットとし，10 個のノードをマップした Chord のリングを示す．SHA1 の値をリング状に並べ，時計回りで測定した長さを距離とする．この距離の定義から，リング状に並べたノード間に近いノードと遠いノードの関係が決まる．ただし，Chord での距離の定義はネットワーク的近さとは無関係である．図 4 では，識別子が 1 のノード N1 から見て N8 や N14 は近いノードとなり，N51 や N56 は遠いノードである．

図 5 に，図 4 のリングにデータをマップした状態を示す．Chord では，識別子空間上にマップするデータの位置情報は，最も近いノードが管理する．図 5 のように，識別子が 24 のデータ K24，識別子が 30 のデータ K30 の位置情報は，リングを時計回りに見て最も近いノード N32 が管理している．

Chord の構成ノードは表 1 に示すように，他のノードへの経路を持つ．識別子 k を与えたとき，時計回りに辿り，最初のノードを返す関数を successor（k）と定義する．successor（k）の返すノードが，識別子 k のデータの位置を格納するといえる．ノードは，サクセッサ・リスト，後述するフィンガー・テーブルの他に，ルックアップの効率化および経路維持のため，predecessor ノードへの経路も持

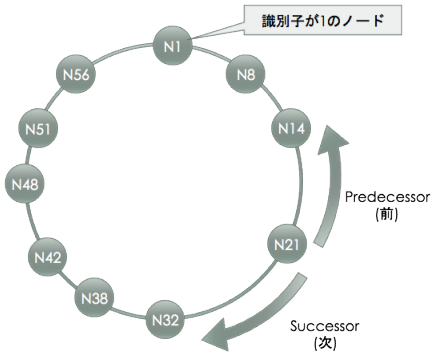


図 4 ノードをマップした Chord のリング (6 ビット識別子空間)

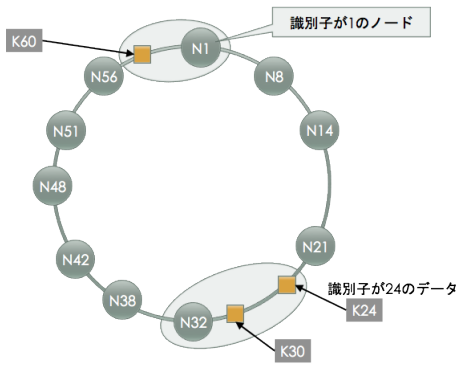


図 5 ノードとデータをマップした Chord のリング (6 ビット識別子空間)

つ．単純化のためサクセッサ・リストの要素数が 1，フィンガー・テーブルがない場合を考える．識別子 k の探索を行う際，ノード自身の識別子 r と識別子 k を比較し，k が r と successor（r）の間であれば，successor（r）が k のデータの位置情報を管理している．k が r と successor（r）の間にない場合，ルックアップ・メッセージを successor（r）のノードへ転送し，転送を繰り返すことでルックアップはいずれ終了する．この場合，ルックアップ・メッセージの転送回数は，ノードの総数に対して線形オーダーとなる．

フィンガー・テーブルは，ルックアップ効率化のために存在する．ノード自身の識別子から 2 の k 乗先のノードへの経路を持つことで，ルックアップ・メッセージを転送するごとに識別子の探索空間の大きさがほぼ半減する．したがって，フィンガー・テーブルを用いると，ノード総数 N に対して $O(logN)$ での探索が可能である．

2.5 CSN

WSN ではパケットロスが発生することが知られている．パケットロスを低減するためには通信距離を短くし，通信強度を高くする必要がある．したがって，Chord におけるノード間距離にネットワーク的近さを考慮しなければならない．CSN は，Chord のリング状識別子空間を階層化することで，センサネットワークに適応させたアルゴリズムである．

階層構造内における階層レベルを 0，1，$\cdots$，m，階層レベル i においてひとつのクラスタを構成する最大ノード数を λ_i，階層レベル i でのノード総数を N_i とする．階層レベル i におけるクラスタ数 φ_i は，式 1 で得られる．

$$\varphi_i = N_i/\lambda_i \tag{1}$$

CSN でのリング作成には，式 2〜式 5 の関係を保証する．したがって，各階層のクラスタ総数は，階層が上がる毎に減少する．これにより，$O(MlogN)$ でのルックアップを可能にしている．(N: 最上層のリング参加ノード数，M: 2 ノード間の最大経路長 +1)．

$$N_0 > N_1 > \cdots > N_k > \cdots > N_m \tag{2}$$

$$\lambda_0 \leq \lambda_1 \leq \cdots \leq \lambda_k \leq \cdots \leq \lambda_m \tag{3}$$

$$N_0/\lambda_0 > N_1/\lambda_1 > \cdots > N_k/\lambda_k > \cdots > N_m/\lambda_m \tag{4}$$

$$\varphi_0 > \varphi_1 > \cdots > \varphi_k > \cdots > \varphi_m \tag{5}$$

3. 実装

最も広く用いられてきた WSN 用オペレーティングシステムである，TinyOS[9] で稼働するよう実装を行った．TinyOS は超小型の WSN ノードを制御するため，専用に開発されたオペレーティングシステムであり，極小な計算資源に対応している．TinyOS ではコンポーネント指向のアーキテクチャを採用し，コンポーネント間の接続は command（コンポーネントの呼び出し），event（割り込み，ハードウェア割り込み）のハンドラを実装することで実現される．

プログラミング言語は，TinyOS 用言語の nesC[10] を用いた．nesC は C の拡張言語であり，コンポーネント指向のプログラミング言語である．プログラマはアプリケーション独自の機能を nesC を用いて記述し，コンポーネントとしてまとめ，TinyOS で標準的に提供されるコンポーネントと接続することでアプリケーションを実現することができる．

3.1 プログラム位置管理機構の構成

図 6 に示すように，プログラム位置管理機構は PacketManager，Neighbors，DHT の 3 つの主要コンポーネントにより構成する．

- PacketManager
  受信したパケットから処理の割り振りを行う Receiver，パケットの送信を行う Sender を内包するコンポーネント．Sender はレスポンス送信の際，パケットの衝突を避けるため，乱数による送信時間の調整を行う．
- Neighbors
  Successor ノードの選定のため，隣接ノードの情報収集を行うコンポーネント．ブロードキャスト通信による周辺ノードへのリクエスト，レスポンスで受け取っ

表 1　Chord のノードがを持つ他ノードへの経路 ([6] より引用)

名称	説明	ルックアップでの役割	維持コスト
サクセッサ・リスト	識別子距離が最も近いノードへの経路．冗長性確保のため，複数ノードへの経路を持つ	サクセッサ・ノードにルックアップを順々に伝搬させてもルックアップ可能（ノード総数に対して線形オーダー）	低い
フィンガー・テーブル	2 の k 乗ずつ遠い識別子距離のノードへの経路（k は 0 以上の整数）	フィンガー・テーブルがすべて正しければ，ノード総数に対して対数オーダーでルックアップ可能	高い

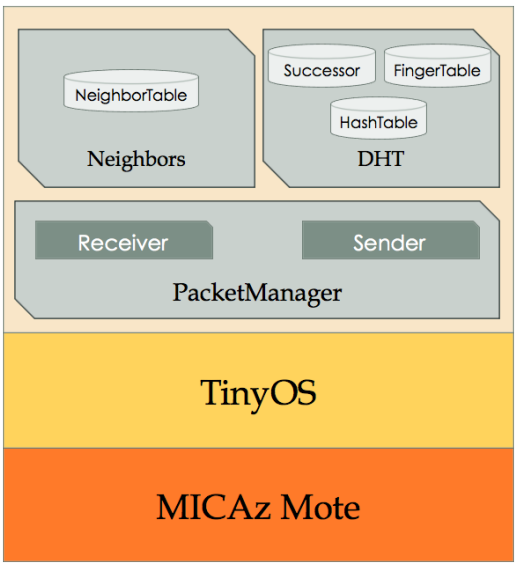
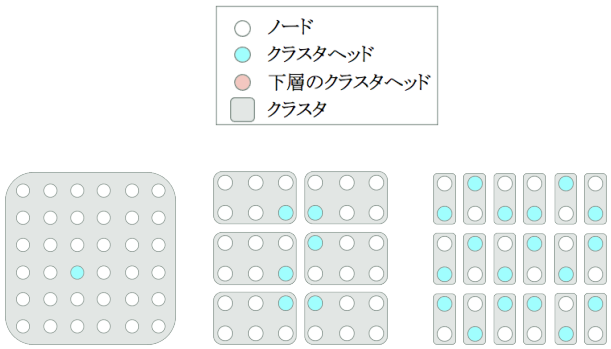


図 6　アーキテクチャ



(a) Level-2 (最上層)　(b) Level-1 (中層)　(c) Level-0 (最下層)

図 7　初期状態

た情報から隣接リストの構築を行う．また，レスポンスで受け取った通信強度の情報を元に，最も近いノードを記録する．

- DHT
  DHT を構築するため，リングの情報を管理するコンポーネント．Successor，Predecessor，FingerTable，HashTable に関わる情報を扱う．

3.2　DHT 構築手順

プログラム位置管理機構では，CSN アルゴリズムを用いて DHT を構築する．センサノードが無線通信を行う性質上，ノード間の距離が重要な要素となる．ゆえに，自律的なクラスタ構築では非効率なクラスタ設定となる可能があるため，DHT を構築する際はクラスタおよび各クラスタヘッドを指定することで，ネットワーク形状に適した識別子リングの形成を補助する．

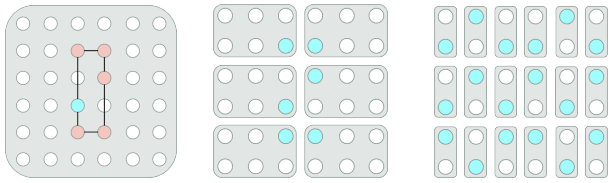
図 7 に，6 × 6 のグリッド上にノードを配備したネットワーク内における，DHT 構築の初期状態を示す．最上層 (a) のクラスタヘッドに DHT 構築の開始命令を送信することで，上層から順次リングの構築を開始する．開始命令を受信したクラスタヘッドは，Neighbors コンポーネントにより周辺ノードのリストを作成する．ここでリストに記録されるノードとは，最下層では自身のクラスタに所属するノード，その他の層ではひとつ下の階層におけるクラスタヘッドである．クラスタヘッドは最も近いノードを Successor ノードとしてリング構築命令を送信し，命令を受けたノードは送信元を Predecessor ノードとして登録する．Predecessor ノードに対して Ack を返した後，クラスタヘッドと同様に Successor ノードの選定を行う．Ack を受けたノードは送信元を Successor ノードとして登録する．この動作を繰り返し，クラスタヘッドの Predecessor ノード登録で最上層リングが構成される．

図 8 に最上層リング構築後の状態を示す．最上層リングの構築が終わると，ひとつ下の階層でのリング構築に入る．リングは最上層と同様に構築を行う．リングをひとつ構築する毎に，上層の Successor ノードへリング構築命令を出すことで，中層すべてのリングが構築される．

最下層でのリング構築は，Neighbors コンポーネントによりクラスタヘッドと同じクラスタ内のノードから隣接ノードが選出される．リングを構築後は中層での処理と同様に，上層の Successor ノードを辿ることで最下層すべてのリングが構築され，図 9 に示すように DHT 構築が完了する．

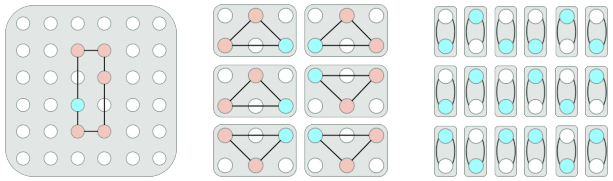
3.3　探索実行手順

探索はすべて最上層から行う．探索実行ノードのクラスタヘッドを辿っていき，最上層リングの参加ノードに対し



(a) Level-2 (最上層)　(b) Level-1 (中層)　(c) Level-0 (最下層)

図 8　最上層リング構築後



(a) Level-2 (最上層)　(b) Level-1 (中層)　(c) Level-0 (最下層)

図 9　DHT 構築完了

リクエストを送信する．リクエストには，探索実行ノード番号およびそのクラスタヘッド，探索対象の SHA1 ハッシュ値を付随する．リクエストを受けた最上層ノードは，自身の担当範囲とリクエストに付随する SHA1 ハッシュ値を比較し，担当範囲でなければ次のノードへリクエストを転送する．これを繰り返すことで DHT 内の探索を実行する．

4. 実験および評価

実装したプログラム位置管理機構の評価を行うため，シミュレーションを行う．シミュレーションにより得られたデータから，本実装の評価を行う．

4.1 実験方法

実験には WSN 用シュミレータである TOSSIM[1] を拡張し，消費電力の測定を可能にした PowerTOSSIM Z[11] を用いる．センサノードの電力は有限であり，電源喪失はネットワークの運用に支障をきたすため，プログラム位置管理機構を実行した際の消費電力を測定する．

図 10 に示すように，シミュレーションは 12×12 のグリッド状にセンサノードを敷設したネットワークを想定して行う．表 2 に，プログラム位置管理機構の構築設定を示す．始めにプログラム位置管理機構の構築時にかかる消費電力を測定する．次に，探索終了までの消費電力を測定し，その結果から構築時にかかる消費量を除き，探索 1 回あたりにかかる消費電力の平均値を算出する．

また，同様のシミュレート環境下における，ランダムウォークでの探索を行った場合の消費電力を測定し，プログラム位置管理機構の測定結果と比較し考察を行う．

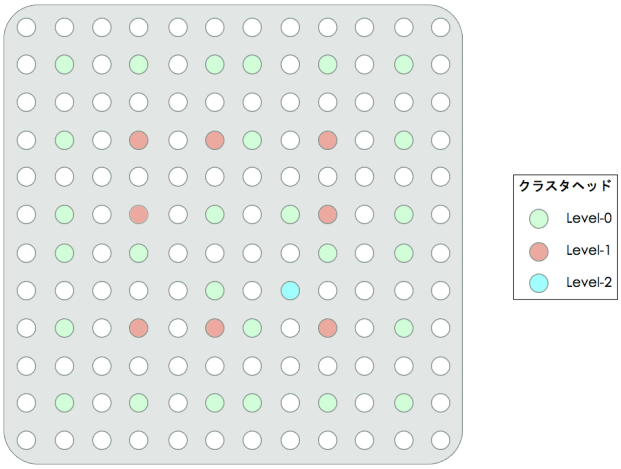


図 10　シミュレーション環境

表 2　プログラム位置管理機構の設定

対象	階層	ノード数
クラスタ数	Level-2	1
	Level-1	9
	Level-0	36
リング参加ノード最大数	Levle-2	9
	Level-1	4
	Level-0	2

4.2 評価

表 3 および表 4 に実験結果を示す．プログラム位置管理機構の消費電力は, 表 3 に示す通りである．センサノードのバッテリ容量が 21,600J であるため，プログラム位置管理機構はバッテリ容量に対して約 0.338%の電力消費で構築可能であることがわかる．また，探索時のバッテリ消費は約 0.013%である．ランダムウォークの場合，一度の探索でバッテリの約 0.033%を消費する．

図 11 に，プログラム位置管理機構とランダムウォークの探索時における消費電力の比較を示す．プログラム位置管理機構は DHT 構築に電力を消費するため，探索回数が少ない場合はランダムウォークに消費電力の点では劣る．しかし，探索回数が 18 回を超えると，探索時の消費電力が低いプログラム位置管理機構の方が全体的な消費電力が低いことが分かる．また，探索回数が 57 回を超えると，プログラム位置管理機構のバッテリ消費約 0.765%に対し，ランダムウォークはその 2 倍以上の 1.872%消費することがわかる．

WSN は一度敷設すると，半年や数年にわたる長期間での運用となるため，プログラムの移動による通信先の変更が多くなると想定される．その結果，プログラム位置管理機構とランダムウォークの探索時における消費電力の差は，非常に大きくなると考えられる．したがって，プログラム位置管理機構における CSN を用いた位置情報の分散管理は，プログラムの位置を探索する際，消費電力を抑えることができるため有効であると考える．

表 3 プログラム位置管理機構の消費電力

対象	消費電力 [J]
DHT 構築	73.03
探索 1 回あたり	2.90

表 4 ランダムウォークの消費電力

対象	消費電力 [J]
探索 1 回あたり	7.10

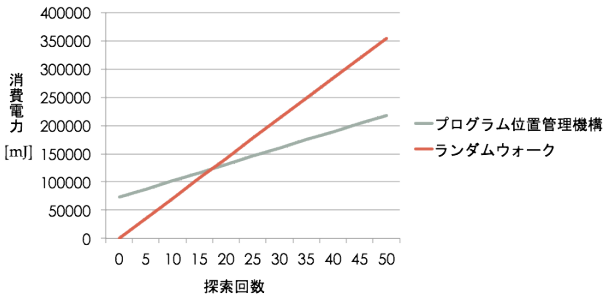


図 11 消費電力の比較

5. まとめと今後の課題

5.1 まとめ

WSN におけるモバイルエージェントアプローチの代表例である Agilla では，プログラム中に通信先のノードを記述する仕様上，WSN 内部を渡り歩くプログラムに対する通信は困難であり，また，その仕様は WSN プログラミングの煩雑さを助長している．そこで，本研究はモバイルエージェントアプローチミドルウェアによるプログラム間通信に起因プログラムの煩雑さを改善するためのミドルウェアで必要となる，プログラム位置管理機構の提案と実装を行った．

プログラム位置管理機構では，DHT を用いてプログラム位置情報を WSN 内に分散管理する手法を採った結果，位置情報の探索にかかる消費電力を抑えることができた．これは，プログラム間通信において通信相手となるプログラムが移動するようなアプリケーションを運用する際，プログラム位置管理機構によるプログラム位置情報の分散管理が，WSN の運用期間を伸ばすことにつながる．したがって，本研究で提案した手法は有効であると考えられる．

5.2 今後の課題

5.2.1 機能拡張

プログラム位置管理機構は，以下の機能拡張を行う必要がある．

- 冗長性確保

  本研究で実装を行ったプログラム位置管理機構には，冗長性が考慮されていない．そのため，DHT を構成するノードが Sleep に入った場合，そのノードが管理する情報を取得することができなくなる．Sleep 処理は WSN 全体の稼働時間を伸ばすことが可能なため，プログラム位置管理機構を Sleep に対応させるべきである．したがって，DHT を構成するノードが管理するプログラム位置情報のバックアップを他のノードに管理させることで，Sleep 時にリングの再構築を行う機能を追加する必要がある．

- 自律的なクラスタヘッド選出

  プログラム位置管理機構を構築するにあたり，クラスタおよびクラスタヘッドに関する情報を入力する必要がある．しかし，この仕様はネットワーク形状への適応を入力情報に依存しているため，ネットワークの大型・複雑化が進むにあたり入力情報が増大することで，ネットワークへの適応が困難になることが予想される．したがって，自律的なクラスタヘッド選出を行うことが解決手法として考えられる．

5.2.2 ミドルウェアへの適応

本研究の最終目的である，モバイルエージェントアプローチによるプログラム間通信に起因するプログラミングの煩雑さ改善を達成するため，本研究で実装したプログラム位置管理機構をミドルウェアへ適応させる必要がある．

参考文献

[1] Philip Levis, Nelson Lee, Matt Welsh, David Culler.: TOSSIM: Accurate and scalable simulation of entire TinyOS applications, In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems*, 2003, pp.126-137.

[2] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong.: TinyDB: an acquisitional query processing system for sensor networks, In *Proceedings of the ACM Transactions on Database Systems*, Vol.30, No.1, 2005, pp.122-173.

[3] T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D.Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. Stankovic, R. Stoleru, A. Wood.: EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks, In *Proceedings of the 24th International Conference on Distributed Computing System*, 2004, pp.582-589.

[4] Chien-Liang Fok, Gruia-Catalin Roman, Chenyang Lu.: Agilla: A Mobile Agent Midleware for Self-Adaptive Wireless Sensor Networks, In *Proceedings of the ACM Transactions on Autonomous and Adaptive System*, Vol.4, No.3, 2009, Article 16.

[5] Gelernter.: Generative Communication in Linda, In *Proceedings of the ACM Transactions on Programming Languages and Systems*, Vol.7, No.1, 1985, pp.80-112.

[6] 江崎浩: “P2P（ピア・ツー・ピア）教科書”，インプレス R&D，(2008-1)

[7] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, Hari Balakrishnan.: Chord: A scalable peer-to-peer lookup service for internet applications, In *SIGCOMM '01 Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, 2001, pp.149-160.

[8] Muneeb Ali, Zartash Afzal Uzmi, LUMS.: CSN: A network protocol for serving dynamic queries in large-scale

wireless sensor networks, In *Proceeding of the Second Annual Conference on Communication Networks and Services Research*, 2004, pp.165-174.

[9] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister.: System architecture directions for networked sensors, In *Proceedings of the ACM SIGPLAN Notices*, Vol.35 No.11, 2000, pp.93-104.

[10] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, David Culler.: The nesC language: A holistic approach to networked embedded systems, In *Proceedings of the ACM SIGPLAN Notices*, Vol.38, No.5, 2003, pp.1-11.

[11] Enrico Perla, Art Ó Catháin, Ricardo Simon Carbajo, Meriel Huggard, Ciarán Mc Goldrick.: PowerTOSSIM z: realistic energy modelling for wireless sensor network environments, In *Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and networks*, 2008, pp.35-42.